## CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Claims 1-143 (Cancelled)

144. (Currently Amended) A method of execution-instruction delegation among multiple elemental processing resources of at least two different types, the multiple elemental processing resources comprising two or more elemental processing resources of a first type each sharing two or more elemental processing resources of a ~~second~~ pipelined computational type with at least one other of the two or more processing resources of the first type, the processing resources of the first type each being ~~defined by a configuration whereby it will~~ configured to receive a non-delegated program instruction from a thread in memory that it can execute or delegate for execution by one computational type elemental processing resource ~~elsewhere~~, the processing resources of the ~~second~~ computational type each being ~~defined by being~~ configured to only receive and execute instructions that were delegated to ~~them~~ their pipeline for execution by processing resources of the first type and by being shared by at least two elemental processing resources of the first type, the method comprising:

obtaining an execution instruction from the thread, wherein the execution instruction is obtained at one of the elemental processing resources of the first type;

determining whether an operation-code within the execution instruction is incapable of being executed by the one elemental processing resource of the first type and thus should be delegated to one elemental processing resource of the ~~second~~ computational type that the one elemental processing resource of the first type shares in common with another elemental processing resource of the first type;

executing the execution instruction with the one elemental processing resource of the first type, if the operation-code within the execution instruction should not be delegated to any elemental processing resource of the ~~second~~ computational type; ~~and~~

if the execution instruction should be delegated, providing a specific elemental processing resource of the first type access to a specific computational type elemental processing resource it shares with another of the elemental processing resources of the first type based upon an arbitration request by the specific elemental processing resource of the first type; and

routing the execution instruction and an identifier for the specific elemental processing resource of the first type that delegated the execution instruction to a pipeline of the specific elemental processing resource of the ~~second~~ computational type that is shared in common and is capable of executing the operation code.

145. (Original) The method of claim 144, wherein the method is completed within a single processing cycle.

Claims 146-154 (Cancelled)

155. (Original) The method of claim 144, wherein the operation-code indicates a type of resource on which to execute.

156. (Previously Presented) The method of claim 144, wherein at least one elemental processing resource of the first type is an originating processing resource.

157. (Previously Presented) The method of claim 144, wherein one or more of the elemental processing resources of the first type is an integer processing unit.

158. (Currently Amended) The method of claim 144, wherein one or more of the elemental processing resources of the ~~second~~ computational type is a mathematical processing unit.

159 (Cancelled)

160. (Currently Amended) The method of claim 144, wherein one or more of the elemental processing resources of the ~~second~~ computational type is a vector processing unit.

Claims 161-163 (Cancelled)

164. (Currently Amended) The method of claim 144, wherein one or more of the elemental processing resources of the ~~second~~ computational type is an execution-instruction processing cache.

165. (Currently Amended) The method of claim 144, further comprising routing the execution instruction through an execution-instruction signal router having arbitration logic that will enable the execution-instruction signal router to arbitrate requests from the elemental processing resources of the first type for access to one of the shared computational type elemental processing resources.

166 (Cancelled)

167. (Currently Amended) The method of claim 144, wherein a first specific elemental processing resource executing a first individual thread may sleep while a second specific shared elemental processing resource executes a delegated execution-instruction[[s]] from the first individual thread.

168. (Previously Presented) The method of claim 144, wherein an execution-instruction signal causes various elemental processing resources dynamically to turn on and off to maintain a desired level of power draw while maximizing processing throughput.

169. (Previously Presented) The method of claim 144, further comprising generating an execution-instruction signal from at least one of the elemental processing resources, wherein the execution-instruction signal from the elemental processing resources themselves shuts off processing resources while idling.

170. (Previously Presented) The method of claim 144, further comprising generating an execution-instruction signal from at least one of the elemental processing resources, wherein an execution-instruction signal from processing resources themselves turn on processing resources when execution-instruction signal processing is required.

171. (Previously Presented) The method of claim 144, wherein the elemental processing resources are communicatively disposed on a same die.

172. (Previously Presented) The method of claim 171, wherein an execution-instruction signal router is on the same die with elemental processing resources.

Claims 173-1614 (Cancelled)

1615. (Currently Amended) A method of execution-instruction delegation among multiple interdependent elemental processing resources, the processing resources comprising ~~multiple elemental processing resources of (a)~~ at least two Instruction Processing Units (IPUs) ~~type processing resources~~ and ~~(b)~~ two or more ~~elemental~~ computational processing resources ~~of an other type~~ comprising, in any combination, one or more of ~~either an~~ MPUs ~~type~~, ~~an~~ instruction delegating caches ~~type~~, ~~an~~ encryption or decryption logic ~~type~~ or [[a]] vector processors ~~type~~, the multiple elemental processing resources being configured to perform processing operations according instructions in an instruction set but being individually incapable of servicing at least one instruction in the instruction set, the elemental processing resources being configured such that a first and a second of the IPUs ~~type processing resources~~ each share at least two of the ~~other type elemental~~ computational processing resources in common, the method comprising:

receiving, from memory, one of multiple execution-instructions of a thread at a first Instruction Processing Unit (IPU) of the at least two IPUs ~~type processing resources~~;

processing the first execution-instruction using the first IPU;

receiving an other execution-instruction of the thread at the first IPU;

determining that the other execution-instruction from the thread can not be processed by the first IPU and must be executed by a computational processing resource;

receiving a request by the first IPU for access to one of the shared-in-common computational processing resources that can execute the other execution instruction;

receiving an acknowledgement that allows delegation by the first IPU to the one of the shared-in-common computational processing resources;

delegating the other execution-instruction from the thread to a pipeline for the one of the two or more ~~elemental~~ shared-in-common computational processing resources ~~of the other type that is shared between the first IPU and a second IPU of the at least two IPU type processing resources, the type of the processing resource other than the IPU type processing resource being such~~ along with an identifier of the first IPU so that it can process the other execution instruction from the thread and return a first result of the execution to the first IPU based upon the identifier;

maintaining the thread in the first IPU in a sleep state until an indicator that the processing of the other execution-instruction from the thread by the ~~other type~~ computational processing resource is returned to the first IPU based upon the identifier;

receiving an execution instruction of a different thread at the second IPU;

determining that the execution-instruction of the different thread can not be processed by the second IPU but can be processed by the one shared-in common computational ~~other type elemental~~ processing resource that is shared between the second IPU and the first IPU;

arbitrating access to the one shared-in-common computational processing resource by the second IPU; and

delegating the execution-instruction from the different thread to the one shared-in-common computational processing resource along with an identifier of the second IPU so that the one shared-in-common computational processing resource can execute the execution-instruction from the different thread and return a second result to the second IPU based upon the identifier of the second IPU ~~other type elemental processing resource that is shared between the second IPU and the first IPU~~.